DEVELOPMENT OF A SPACE-TIME WINDOW FOR LARGE EDDY SIMULATIONS

Thesis

Presented in Partial Fulfillment of the Requirements for Graduation with Honors Research Distinction in the College of Engineering of The Ohio State University

By

Brian C. Munguía

Undergraduate Program in Aeronautical and Astronautical Engineering

The Ohio State University

2014

Honors Thesis Committee:

Professor Datta Gaitonde, Advisor

Professor Mei Zhuang

Abstract

A major problem in the field of aeronautics is jet noise. Jet noise is caused by turbulence in a flow and is still difficult to predict even with the use of computational techniques. The difficulty arises from determining the location of the source of noise in a specified region. The proposed research will analyze jet flows using window functions. Window functions are mathematical functions that have a value of zero outside of a specified interval. Applying window functions to a simulation allows isolation of specific regions in a flow. The High-Fidelity Computational Multi-Physics Laboratory, or HFCMPL, already has datasets for various flow regimes as well as codes to analyze the data. The goal of the project is to develop a function that will analyze the effects of a small, localized disturbance in a jet flow, allowing better correlation between local phenomena and their effects downstream. The function will be implemented into the existing codes to gain further understanding of jet noise.

Acknowledgements

I would like to thank my research advisor, Professor Datta Gaitonde, for giving me the opportunity to pursue this research as well as the support and guidance that he provided along the way. He not only invested time and resources that helped my development as an aerospace engineer, but he has also helped motivate me to pursue a graduate education and further grow as a researcher.

I would also like to thank Professor Mei Zhuang for the impact she has had on my undergraduate career. From the instruction she provided in the classroom to serving as a member of my honors thesis committee, I truly appreciate everything she has done for me.

My fellow students were also instrumental in my success. I would like to thank the graduate students from my lab, especially Dr. Robert Yentsch, Mbu Waindim, Rachelle Speth, and Kalyan Goparaju for the assistance they have provided with my research. I also appreciate my classmates, especially Nick Bui, Ben Lewis, and Dan Richie for their hard work and the many sleepless nights we spent in Scott Lab.

Nobody has been as supportive of my education as my parents, José and Denise Munguía. Without their support and encouragement, I would not be where I am today.

Finally, I would like to thank the College of Engineering for the financial assistance they have provided in support of my research.

Vita

April 14, 1992 ……Born, North Olmsted, Ohio, USA

May 4, 2014 ……B.S., Aeronautical and Astonautical Engineering, The Ohio State University

Field of Study

Major Field: Aeronautical and Astronautical Engineering

Table of Contents

List of Figures

1. Introduction

The field of aeroacoustics is now over 60 years old and jet noise research, an important branch of aeroacoustics, has seen significant progress over that time. With each new decade came new theories about the cause of jet noise. While the earliest theories formed in the fifties stated that small eddies caused jet noise, theories from the eighties believed that the cause was large-scale eddies. Modern jet noise researchers generally agree that the cause is actually a combination of both small- and large-scale turbulence structures and how they interact. Although understanding both is important, the proposed research will focus mainly on the large-scale turbulence structures. The recent use of computer simulation has allowed the field of jet noise research to advance more rapidly than ever before. Even after so much progress has been made, our understanding of turbulence is still rather limited.

The expected outcome of the research is to develop a tool to determine the source of jet noise for a given location in the far field through the use of window functions. Datasets with velocity, pressure, and density data for various flow regimes, both with and without the use of plasma actuators, have already been created, so the proposed research only requires the development of the window functions to further analyze the datasets.

While the main objective of the project will be to use windows to analyze jet noise, the use of windows has applications in other areas, such as analysis of stalled wing sections. So another objective is to ideally create a code which is universal so it can be implemented into various codes.

2. Background

2.1 Jet Noise

The earliest theories concerning jet noise originated in the fifties. These theories used the compressible equations of motion to represent the propagation of acoustic waves, as shown in Equation 1:

$$\frac{\partial^2 \rho}{\partial t^2} - a_\infty^2 \nabla^2 \rho = \frac{\partial^2 T_{ij}}{\partial x_i \partial x_j} \tag{1}$$

where ρ is the density of the fluid, a_∞ is the speed of sound of the freestream, and T_{ij} is the stress tensor. The left side of Equation 1 describes acoustic wave propagation, and the right side of the equation describes the noise sources, also known as quadrupoles. These early theories believed that jet noise was caused by random, small eddies that were in some way related to the source term [1].

While no formal relationship was established between these small eddies and noise source terms, these early theories did give good insight into how noise propagates in a jet. It was theorized and verified that noise sources are convected downstream by the mean flow, and that the effects of this convection are more pronounced at higher flow speeds. Furthermore, the highly nonuniform nature of the mean flow causes the noise to refract as it moves through the jet, as shown in Figure 1.

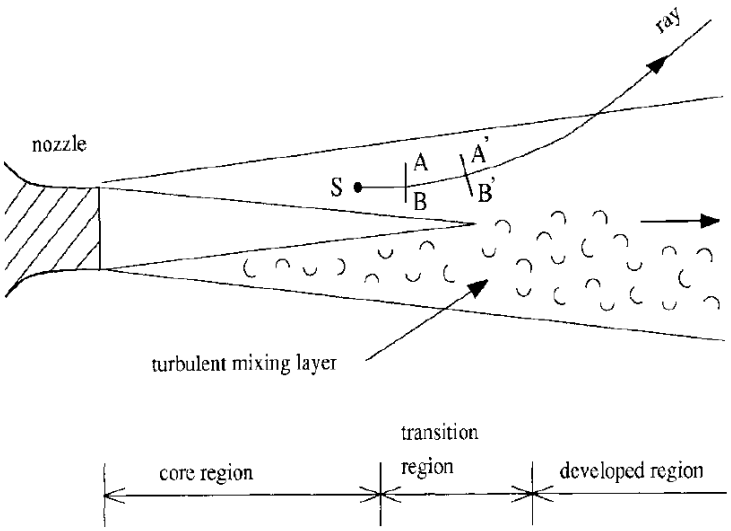

Figure 1: Refraction of noise in jet mixing layer [1]

Assuming a constant temperature within the flow and thus, a constant speed of sound, it is intuitive that the flow velocity at Point B is higher than at Point A. This causes refraction as noise propagates downstream from the source at Point S [1].

2.2 Noise Control

The relatively recent field of computational fluid dynamics, or CFD, has helped jet noise research see large strides in recent years. From analysis, it can be shown that disturbances in a jet are affected by a combination of fluid velocity, pressure, and density. It is understood that the interaction of turbulence structures, which are a result of these disturbances, create jet noise, as mentioned previously. With this understanding, several solutions have been proposed to mitigate jet noise. One such solution currently under investigation is active flow control through the use of localized arc filament plasma actuators. Research indicates that turbulence structures in jets have responded to a large

range of forcing frequencies from plasma actuators [2]. Computational models have also shown this response, as seen in Figure 2.

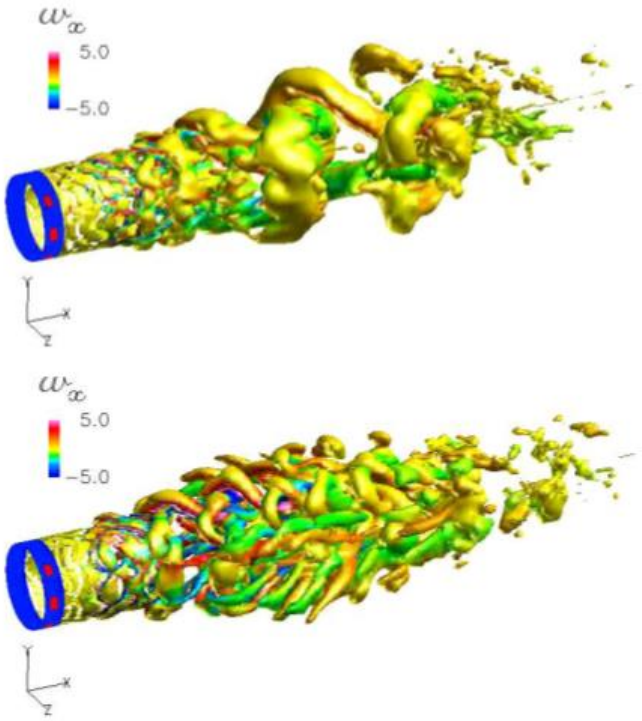

Figure 2: LAFPA controlled jets at St=0.3 (top) and St = 0.6 (bottom) [3]

The images depicted in Figure 1 are vorticity contours of a Mach 1.3 jet being controlled by Localized Arc Filament Plasma Actuators at different forcing frequencies. The frequencies are given by the non-dimensional number, Strouhal number:

$$St = \frac{fD}{U} \tag{2}$$

where St is Strouhal number, f is the forcing frequency, D is the length scale, and U is the velocity scale. For the image shown, D is defined as the diameter of the jet and U is the jet velocity. The top image is at St=0.3, while the bottom image is at St=0.6. It can be seen that the higher frequency forcing does a better job of breaking up the large-scale turbulent structures, and that there are more hairpin-like structures [3].

2.3 Window Functions

Although there is definitely an observable response, the understanding of the response is still tenuous. The problem arises not from understanding the cause of jet noise, but rather, from being able to locate its source. Determining the source of noise is very complicated, and more complications arise with a complex flow such as that of a jet with multiple scales of turbulence structures. The ability to know which region of a jet generates noise in a certain location in the far field will allow for better local jet control. This research will isolate regions of the jet during simulation in order to determine their effect on noise production.

Regions of a simulated flow can be isolated through the use of window functions. Window functions are mathematical functions which has a value of zero outside of a specified interval. Ideally, window functions are smooth due to the undesirable effects which can result from discontinuity. Several window functions already exist, including rectangular windows and generalized cosine windows such as the Hann window and the Hamming window. The Hann window is given by

$$w(n) = 0.5\left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right) \tag{3}$$

where n is the index of the point within the window and N is the total number of points within the window. The Hamming window is given by

$$w(n) = \alpha - \beta\cos\left(\frac{2\pi n}{N-1}\right) \tag{4}$$

where $\alpha \approx 0.54$ and $\beta \approx 0.46$. Plots of the Hann and Hamming windows can be seen in Figures 3 and 4, respectively.

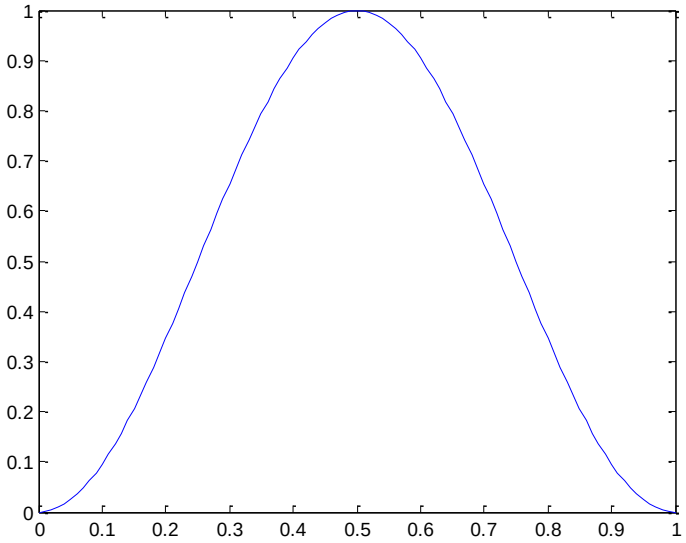

Figure 3: MATLAB plot of Hann window

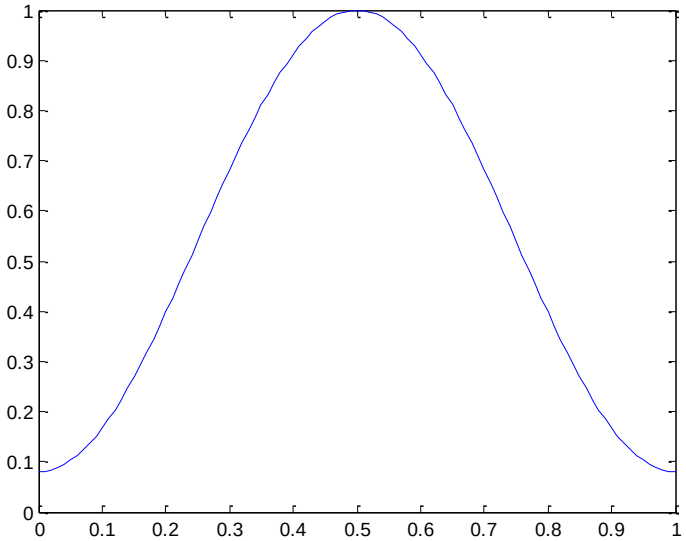

Figure 4: MATLAB plot of Hamming window

HFCMPL currently uses window functions to analyze jet noise propagation. By isolating a region of a flow and subtracting the mean velocity from the velocity in the region, velocity of the disturbance can be determined and, thus, the speed at which pressure waves propagate from that region. However, the current code utilizes

rectangular windows which, as stated earlier, can result in undesirable effects when post-processing data.

2.4 Fast Fourier Transform

The Fast Fourier Transform (FFT) of a signal is a method of extracting a series of sine and cosine functions to represent the signal. The time domain signal is treated as a periodic function, and a frequency domain signal is extracted from this function. The resulting frequency domain signal is a series of peaks at various frequencies. The frequencies represent the frequencies of the sine and cosine functions, and the magnitudes of the peaks at these frequencies represent the coefficients of the sine and cosine functions, or the strengths of these sine and cosine functions in the signal.

The FFT of a signal can be calculated by first decomposing an N-point time domain signal into N separate time domain signals. Frequency spectra are then calculated using these signals, and the spectra are combined to form a single frequency spectrum [4].

The Power Spectral Density (PSD) is a very useful tool for understanding stationary signals. The PSD of a signal describes the power distribution of a time domain signal over the frequency spectrum. The PSD is defined as

$$S_{xx} = \lim_{T\to\infty} E[|\hat{x}_T(\omega)|^2] \tag{5}$$

where $\hat{x}_T(\omega)$ is the FFT of the time domain signal, and E is the expected value. As can be seen by Equation 5, the FFT of the signal is first calculated, and power can be obtained by squaring the absolute value of the FFT. The signal can be divided by sampling frequency and length of the signal to obtain unites of power/Hz, then multiplied by 2 to account for negative frequencies [5].

A typical power spectrum of a turbulent flow is characterized by an energy cascade. The large-scale, lower frequency turbulence structures are much stronger than the small-scale, higher frequency structures. This energy cascade is still evident with the use of plasma actuators, as is shown in Figure 5 below.

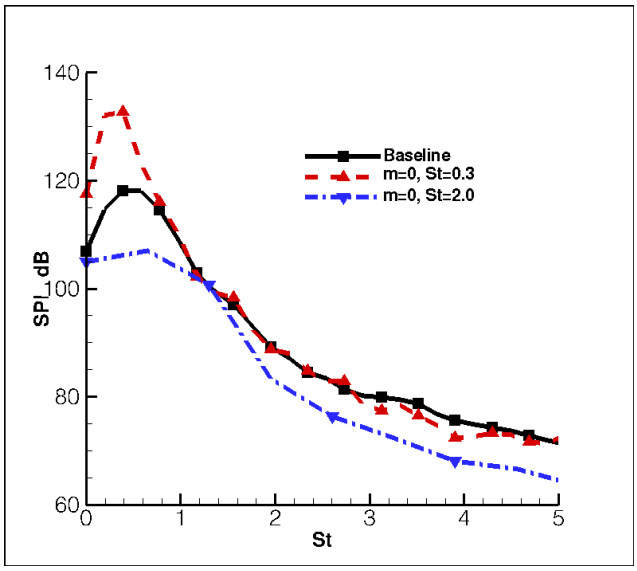

Figure 5: Power spectrum of a turbulent jet flow with and without LAFPAs [3]

The datasets analyzed in this research are taken from simulations of turbulent jet flow and, thus, should exhibit a similar trend in the frequency domain.

3. Methodology

The general research plan of the project is to develop a code that can analyze the effects of a disturbance in a flow simulation using various window functions, then to analyze the frequency characteristics of these different windows. The current analysis code used by the HFCMPL utilizes rectangular windows to analyze disturbances, but these windows are harsh and have adverse effects on the frequency characteristics of the signal. Due to these adverse effects, this project focused on the use of generalized cosine functions which do not have abrupt changes. Specifically, the functions used were the Hann and Hamming windows described by Equations 3 and 4, respectively, as well as a custom window function. The custom function is an altered version of the Hann window and will be referred to as the modified Hann window. The function rises as a cosine function for 1/3 the window length, has a value of 1 for 1/3 the window length, then finally decays as a cosine function for the remainder of the window length. The equation is given by the piecewise function in Equation 6.

$$w(n) = \begin{cases} 0.5\left(1 - \cos\left(\frac{3\pi n}{N-1}\right)\right) & if\ \frac{n}{N} \leq \frac{1}{3} \\ 1 & if\ \frac{1}{3} < \frac{n}{N} < \frac{2}{3} \\ 0.5\left(1 + \cos\left(\frac{3\pi n}{N-1}\right)\right) & if\ \frac{n}{N} \geq \frac{2}{3} \end{cases} \tag{6}$$

The function is depicted in Figure 6 below.

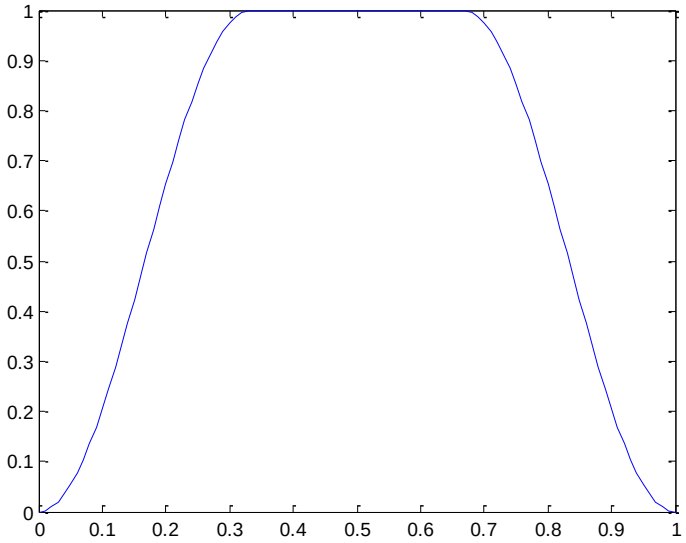

Figure 6: MATLAB plot of modified Hann window

The purpose of developing this window was to create a window with a value of 1 over a wide range of values. Existing windows functions, excluding the rectangular window, only have a value of 1 at a single point within the window. This would limit the analysis of the disturbance to a single point in space or time. The modified Hann window, on the other hand, can be used to analyze a disturbance within a specified region without the adverse effects of sharp corners as seen in the rectangular window. The research will determine the effects of "stretching" the windowed region by comparing the results from the modified Hann window with those from the Hann and Hamming windows.

In order to apply these window functions to a flow, they must first be converted from 1-D functions to 4-D functions. In 3-D space, the functions were converted to functions of radius from a center point to make the windows independent of orientation. Thus, the function would be some window W(x,y,z)=w(r) where

$r = \sqrt{(x - X_c)^2 + (y - Y_c)^2 + (z - Z_c)^2}$ for a window centered at (X_C,Y_C,Z_C).

Once the space window was generated, a time window would be multiplied by this window such that w(r,t)=w(r)w(t). An example of converting a window is shown in Figure 7.

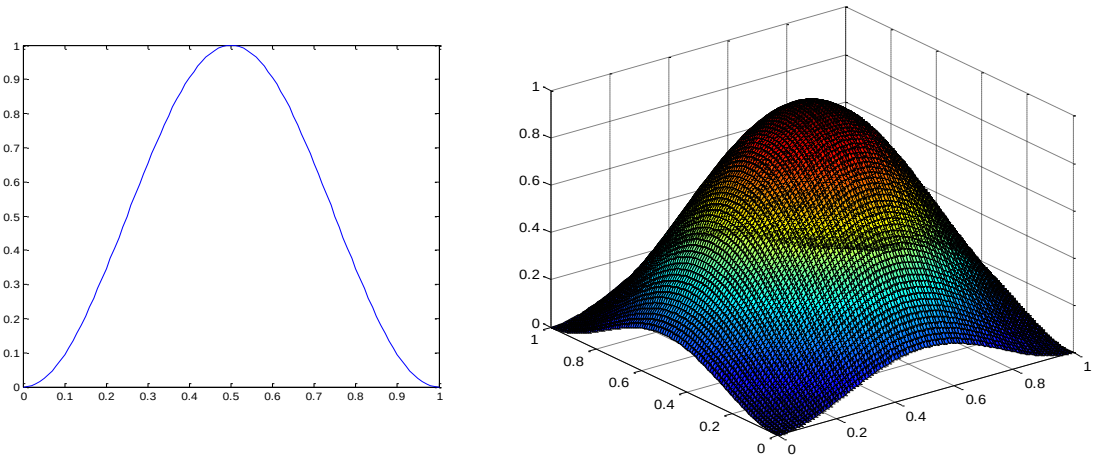

Figure 7: Conversion of Hann window from 1-D to 2-D

The window shown is a Hann window centered at x=0.5 in 1-D and (x,y)=(0.5,0.5) in 2-D, and has a radius of 0.5. It can be seen that the function tapers off to 0 in the 2-D function just as it did in 1-D.

Visualizing the 3-D and 4-D equivalents of these window functions can be difficult. The 3-D window function would be a sphere with a value of 1 at the center that tapers off by a cosine function until it reaches 0, or some small value in the case of the Hamming window, at the edges. The 4-D window would then be this 3-D window that varies with time. It would start with a value of 0 everywhere, then the windowed region would ramp up in value over time until the time window reaches its maximum value, at which point the window will decay back to 0.

The window code, which can be found in the Appendix, was then applied to existing datasets to determine the suitability of the different functions for use in simulation. Each dataset was a movie file at and represented a single timestep, so the

window function was called in a loop. A center point, radius, and time interval were specified, the datasets were read, windowed based on the specifications, and new data files were created. These files were then read into FieldView 13 to create animations. Images of the windowed datasets will be presented in the Results and Discussion section of the report.

After applying the window code to the datasets, the next step was to calculate the PSDs of the datasets for comparison. The FFT code used was based off of the code provided by Dr. Smith in "The Scientist and Engineer's Guide to Digital Signal Processing." The code would first decompose the N-point signal into N single point signals via bit reversal data sorting. The code would then calculate the frequency spectrum of each point. Finally, the N frequency spectra are combined to form a single frequency spectrum [4]. The FFT code used can be found in the Appendix. Calculation of the PSD was performed in MATLAB using the manipulations described in Chapter 2.

4. Results and Discussion

The window functions were applied to an existing data set for a Mach 1.3 jet as shown in Figure 8. The grid for the simulation was 211×121×53, ranging from x = 0 to x = 30.39, y = -10.20 to y = 10.20, and z = -10.18 to z = 10.18. Each window was centered at (x,y,z)=(15,0,0) and had a radius of 1. The images displayed show contours of normalized pressure, which was the parameter assessed in this research. Jet noise is affected by a combination of fluid pressure, velocity, and density, so analyzing any of these parameters would yield similar results.

It can be seen in the figure that in the windowed regions, the pressure is maximized inside the region, indicated by red, and tapers off to zero at the edge of the region, indicated by blue. This acted as a sanity check that the window function worked properly.

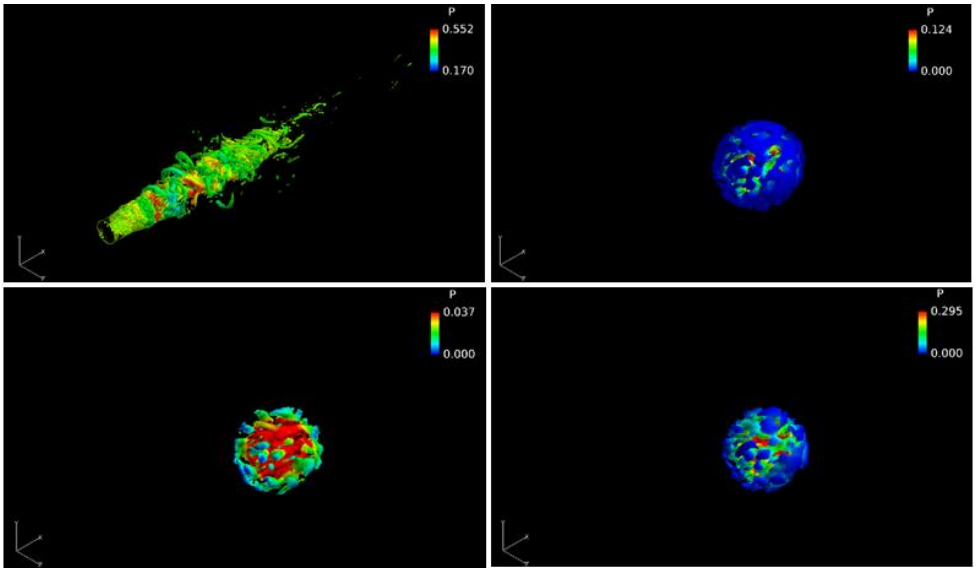

Figure 8: Pressure contours of unwindowed (top left), Hann (top right), Hamming (bottom left), and modified Hann windowed (bottom right) datasets

The datasets were not only windowed in space as shown in the image above, but also in time. Each dataset corresponded to a different timestep with a sampling rate of 0.005 s. There were 99 datasets used in the research, or 0.495 s of data.

After windowing, the average value of pressure was taken within the windowed region and the PSD was calculated based on the variation of the average value over time. Results of the PSD calculation can be found in Figure 9 below. The top left image shows the PSD for the unwindowed case, which exhibits the energy cascade that is characteristic of turbulent flows. The top right and bottom left images show the PSDs after the application of the Hann and Hamming windows, respectively. It can be seen that the roll-off of energy is much steeper for the Hann and Hamming window cases. This seems to

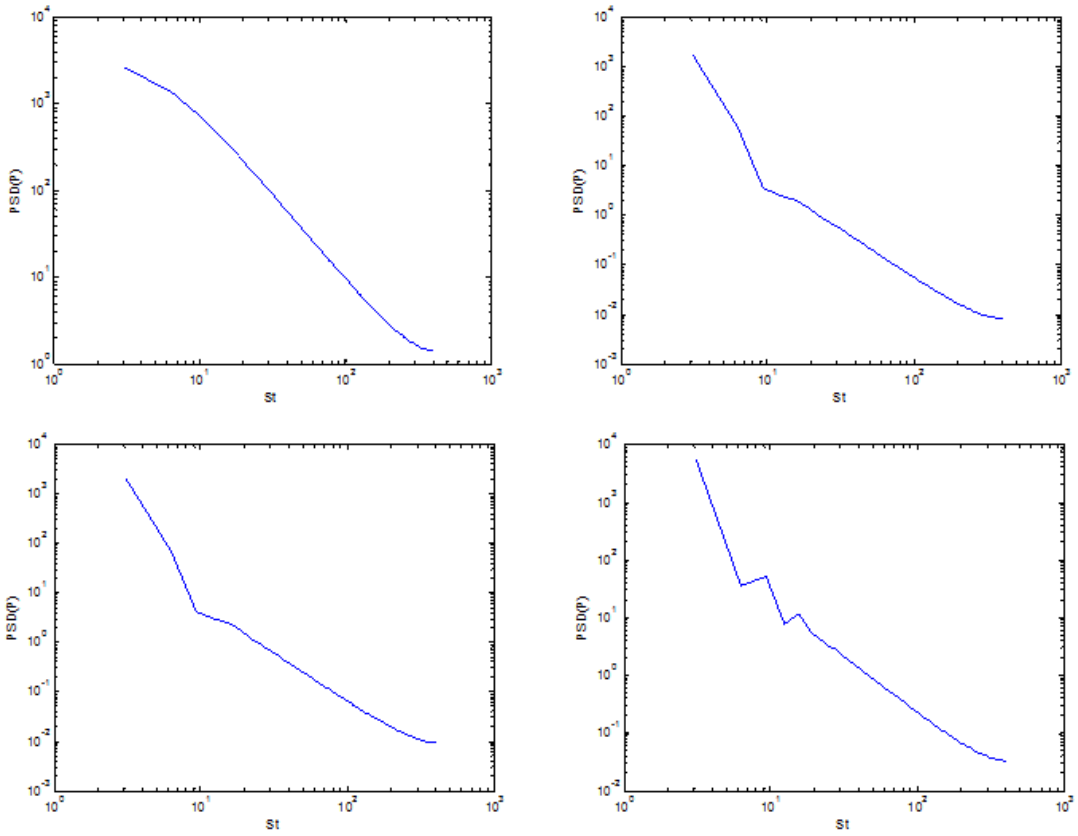

Figure 9: Power spectral density of 4-D unwindowed (top left), Hann windowed (top right), Hamming windowed (bottom left), and modified Hann windowed (bottom right) pressure data

indicate that the application of the window, which is a low frequency signal, dominated the signal and suppressed the higher frequencies.

The application of the modified Hann window, however, had the most interesting effect on the power spectrum. The PSD plot shows a few sharp features, indicating either prominent peaks that did not exist before the application of the window, or issues resolving certain frequencies. Either way, the issue was most likely due to the use of the modified Hann window in the time domain, and a 1-D study was performed to confirm this.

In the study, data at a single point was windowed over time and the PSD of this data was calculated. The dataset used in the 1-D study was much larger, consisting of 120000 timesteps as opposed to the 99 timesteps in the previous 4-D study. This was to ensure that the issues encountered were not caused by poor frequency resolution. Plots of PSD from this study are shown in Figure 10. The results indicate that the Hann and Hamming windows had very little qualitative effect on the power spectrum, but the magnitudes did vary from the unwindowed case.

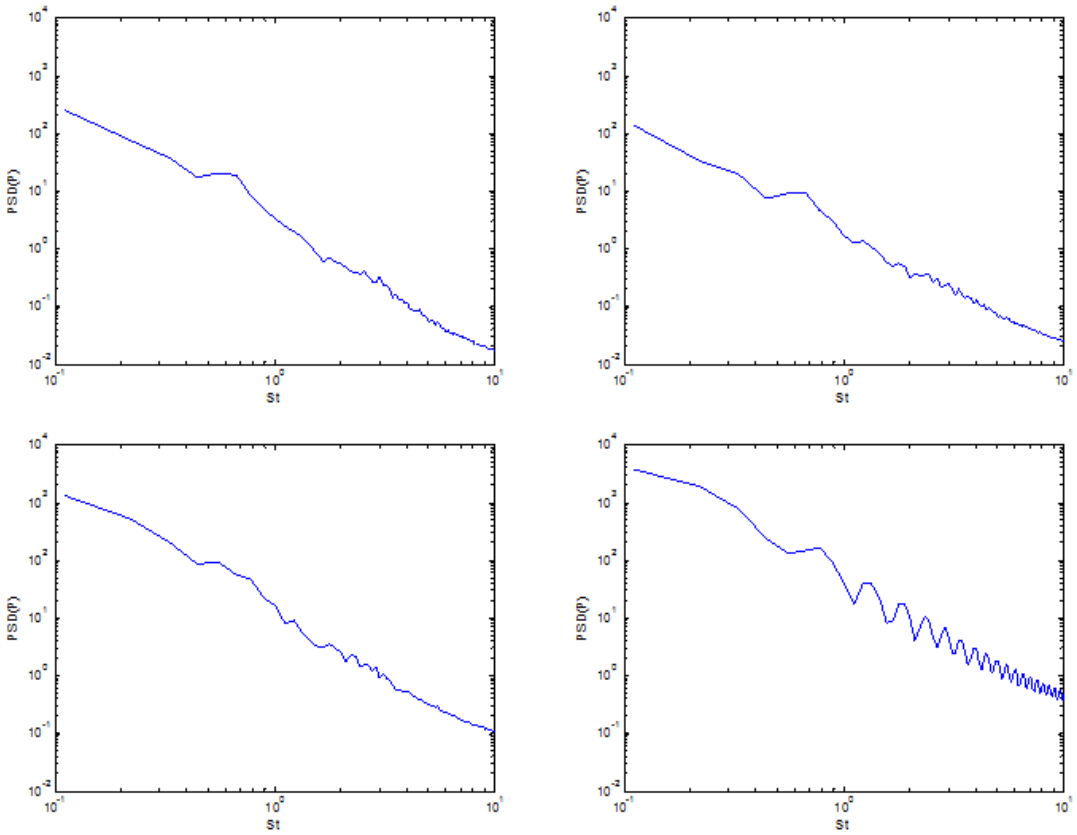

Figure 10: Power spectral density of 1-D unwindowed (top left), Hann windowed (top right), Hamming windowed (bottom left), and modified Hann windowed (bottom right) pressure data

The modified Hann window once again had adverse effects on the frequency spectrum of the signal. The spectrum had several "dips" at the higher frequency, as is seen when trying to perform an FFT on a square wave or rectangular window. The modified Hann window has smooth transitions so, unlike with a rectangular window, the issue is not with sharpness. The only modification to the Hann window is that the middle portion of the window is stretched, so it is apparent that this flat region has some adverse effect on the frequency spectrum.

5. Conclusions

In this research, a computational tool was developed for the purpose of studying jet noise in large eddy simulations. The tool was analyzed to assess the viability of the use of different window functions for analyzing small disturbances in a flow simulation. Three different window functions were studied: the Hann window, the Hamming window, and a modified Hann window.

Analysis of the different windows indicated that the Hann and Hamming windows were most suitable for windowing in time. Both of these windows had no qualitative effect on the frequency spectrum in the 1-D study. Therefore, if a disturbance was analyzed using a Hann or Hamming window in time, one could expect to see a similar power spectrum as with the actual, unwindowed signal.

The modified Hann window had issues resolving higher frequencies in the FFT, which is due to the stretched region of the window. The FFT is a method of extracting a series of sine and cosine functions from a signal, which is difficult to do in the presence of sharp corners or a constant amplitude signal.

While the modified Hann window is unsuitable for time windowing, further investigation is required to determine how useful it is for space windowing. This will require modification to the window code. In its current state, the code can only generate one type of window in space and time. The code will be modified to allow the use of different windows on a dataset.

The Hann and Hamming windows showed no adverse effects in the 1-D analysis, but they were qualitatively different than the unwindowed case in the 4-D analysis. Future work will look into the effects of different parameters of the window, such as the length of the window in space and time, to further improve the capabilities of the window code.

References

[1] Tam, C.K.W. “Jet Noise: Since 1952.” Theoretical and Computational Fluid Dynamics, Volume 10, Issue 1-4, 1998, pp. 393-405.

[2] Samimy, M., Kim, J.H., Kastner, J., Adamovich, I., Utkin, Y. “Active control of high-speed and high-Reynolds-number jets using plasma actuators.” Journal of Fluid Mechanics, Volume 578, 2007, pp. 305-330.

[3] Speth, R.L. and Gaitonde, D.V. “The Effect of Laminar Nozzle Exit Boundary Layer Thickness on a Mach 1.3 Jet With and Without Control.” AIAA Fluid Dynamics Conference, New Orleans, LA, June 2012.

[4] Smith, Steven W. “The Scientist and Engineer’s Guide to Digital Signal Processing.” California Technical Publishing, 2011, pp. 225-242.

[5] Oppenheim, Alan V. and Verghese, George C. “Signals, Systems, and Interference.” MIT OpenCourseWare, 2010, pp. 183-193.

Appendix

xyztwin_module: calculates window amplitude based on Cartesian coordinates of point, time step, and specified window parameters

```
module xyztwin_module
!Generates space time window function
implicit none
contains
  subroutine xyztamp(x,y,z,t,il,jl,kl,xyzc,Rwin,tsta,tend,
wintype,winamp)

    ! INPUT:
    !  x,y,z    - arrays containing values to be windowed
    !  t        - timestep of current dataset
    !  il,jl,kl - lengths of i,j,k vectors
    !  xyzc     - cartesian coordinates of center of window
    !  Rwin     - radius of window
    !  tsta     - start of time window
    !  tend     - end of time window
    !  wintype  - int specifying window type
    ! OUTPUT:
    !  winamp   - array containing window amplitude values

    real*4, intent(in) :: x(:,:,:), y(:,:,:), z(:,:,:),
xyzc(3), Rwin, t, tsta, tend
    real*4, intent(out) :: winamp(:,:,:)
    integer, intent(in) :: wintype, il, jl, kl
    real*4 :: tinwin, tamp, Ltwin, xyzinwin, xyzamp, xc,
yc, zc, pi, a, B
    integer :: i,j,k

    pi=4.*atan(1.)
    Ltwin=tend-tsta

    xc=xyzc(1)
    yc=xyzc(2)
```

```
    zc=xyzc(3)

! Hann window (wintype==1)
    if(wintype==1)then
       do i=1,il
          do j=1,jl
             do k=1,kl
                tinwin=t-tsta
! If dataset is inside time window
                if(tinwin>=0.and.tinwin<=Ltwin)then
                   tamp=0.5*(1.-cos(2.*pi*tinwin/Ltwin))
                   xyzinwin=Rwin-sqrt((x(i,j,k)-
xyzc(1))**2+(y(i,j,k)-xyzc(2))**2+(z(i,j,k)-xyzc(3))**2)
! If point is inside space window
                   if(xyzinwin>=0)then
                      xyzamp=0.5*(1.-
cos(pi*xyzinwin/(Rwin)))
                   else
                      xyzamp=0
                   endif
                   winamp(i,j,k)=tamp*xyzamp
                else
                   winamp(i,j,k)=0
                endif
             enddo
          enddo
       enddo
       return

! Hamming window (wintype==2)
    elseif(wintype==2)then
       a=25./46.
       B=1.-a
       do i=1,il
          do j=1,jl
             do k=1,kl
                tinwin=t-tsta
! If dataset is inside time window
                if(tinwin>=0.and.tinwin<=Ltwin)then
                   tamp=a-B*cos(2.*pi*tinwin/Ltwin)
                else
```

```
                  tamp=a-B
               endif
               xyzinwin=Rwin-sqrt((x(i,j,k)-
xyzc(1))**2+(y(i,j,k)-xyzc(2))**2+(z(i,j,k)-xyzc(3))**2)
! If point is inside space window
               if(xyzinwin>=0)then
                  xyzamp=a-B*cos(pi*xyzinwin/Rwin)
               else
                     xyzamp=0
               endif
               winamp(i,j,k)=tamp*xyzamp
            enddo
         enddo
      enddo
      return

! Custom window (Modified Hann) (wintype==3)
    elseif(wintype==3)then
      do i=1,il
         do j=1,jl
            do k=1,kl
               tinwin=t-tsta
               xyzinwin=Rwin-sqrt((x(i,j,k)-
xyzc(1))**2+(y(i,j,k)-xyzc(2))**2+(z(i,j,k)-xyzc(3))**2)
! If dataset is inside time window
               if(tinwin>=0.and.tinwin<=Ltwin)then
                  if(tinwin/Ltwin<=1./3.)then
                     tamp=0.5*(1.-cos(3.*pi*tinwin/Ltwin))
                  elseif(tinwin/Ltwin>=2./3.)then
                     tamp=0.5*(1.-cos(3.*pi*(Ltwin-tinwin)
/Ltwin))
                  else
                     tamp=1.
                  endif
! If point is inside space window
                  if(xyzinwin>=0)then
                     if(xyzinwin/Rwin<=2./3.)then
                        xyzamp=0.5*(1.-
cos(1.5*pi*xyzinwin/Rwin))
                     else
                        xyzamp=1.
```

```
                        endif
                     else
                        xyzamp=0
                     endif
                     winamp(i,j,k)=tamp*xyzamp
                  else
                     winamp(i,j,k)=0
                  endif
               enddo
            enddo
         enddo
         return

!Rectangular window (wintype==4)
      elseif(wintype==4)then
         do i=1,il
            do j=1,jl
               do k=1,kl
                  tinwin=t-tsta
! If dataset is inside time window
                  if(tinwin>=0.and.tinwin<=Ltwin)then
                     tamp=1.
                     xyzinwin=Rwin-sqrt((x(i,j,k)-
xyzc(1))**2+(y(i,j,k)-xyzc(2))**2+(z(i,j,k)-xyzc(3))**2)
! If point is inside space window
                     if(xyzinwin>=0)then
                        xyzamp=1.
                     else
                        xyzamp=0.
                     endif
                     winamp(i,j,k)=tamp*xyzamp
                  else
                     winamp(i,j,k)=0.
                  endif
               enddo
            enddo
         enddo
         return
      endif
   end subroutine
end module xyztwin_module
```

fft_module: given a dataset of length M, calculates fast Fourier transform of length N

```
module fft_module

!Calculates FFT of a complex dataset

implicit none
contains
  subroutine fft(x,M)
    real :: pi
    integer :: M, N, i, j, k, L, ip
    complex :: x(:), u, w, t

!Set constants
    N=2**M
    j=1
    pi=4.*atan(1.)

!Bit reversal sorting
    do i=1,N-1
       if(i<j)then
          t=x(j)
          x(j)=x(i)
          x(i)=t
       endif
       k=N/2
       if(k<j)then
          do while(k<j)
             j=j-k
             k=k/2
          enddo
       endif
       j=j+k
    enddo

    do k=1,M
       L=2**(M+1-k)
       u=(1.0,0.0)
!Calculate sine and cosine values
       w=cmplx(cos(2.*pi/real(L)),-sin(2.*pi/real(L)))
!Loop for each DFT
       do j=1,L/2
```

```
!Loop for each butterfly calculation
          do i=j,N,L
!Butterfly calculation
             ip=i+L/2
             t=x(i)+x(ip)
             x(ip)=(x(i)-x(ip))*u
             x(i)=t
          enddo
          u=u*w
       enddo
    enddo
    return
  end subroutine fft
end module fft_module
```